Human → Δ (input)

Δ ⟼ { λ_q(Δ) , λ_s(Δ) } (dual targets)

λ_q : Δ → $QASM_3$ (quantum gate list)

λ_s : Δ → $\mathbb{C}^{2^n \times 2^n}$, n ≲ 45 (state-matrix sim; n≈qubits)

Σ = { ⊗, ⊕, ·, †, Πθ, e^{iθ}, H,X,Y,Z,Rx(θ),U(θ,φ,λ), |b⟩, ⟨b|, M_z[q] }

Grammar G = (N,Σ, P, S_0)

P :

S_0 → ι = E

E → E ⊗ E | E · E | G_0 | K | B

G_0 → H | X | Y | Z | Rx(θ) | U(θ,φ,λ)

K → |b⟩

B → ⟨b|

Example

ψ_0 = |0⟩ ⊗ |0⟩

U = H ⊗ X · Rx(π/8)

ψ_1 = U · ψ_0

M_z = ⟨0| ⊗ I · ψ_1

Wave-primitives

Ω_f (continuous drive, freq f)

η(ℛ) (QRNG stream ℛ → phase)

⊡F{E} (Fourier block on expression E)

Provenance

hash = SHA-256(Δ ‖ seed)

deterministic↔random : η(ℛ) ⇄ η(PRNG)

Why a Wave-Native Programming Notation?

Delta-Intermediate Representation (Δ-IR) is a pocket-sized, wave-native programming notation in which every line *is* linear algebra: Dirac "bra–ket" vectors, tensor products, and phase operators. Source files compile directly to (a) Open Quantum Assembly Language version 3 for quantum-hardware runtimes **or** (b) a high-performance classical simulator. There are no English keywords and no Python translation layer—only mathematics. Designed by musicians, clinicians, and quantum hobbyists, Δ-IR lets practitioners treat computation the way a sound-engineer treats audio: shaping phase, interference, and entropy in real time.

```
Human  →  Python  →  Qiskit / Cirq / Q-sharp  →  Open QASM  →  hardware    (today)

Human  →  Δ-IR (pure math)  →  tensor graph  →  Open QASM / numerical kernels   (Δ-IR path)
```

Mission: *Use waves—not prose—to improve minds and advance quantum software.*

Core Symbols and Mini-Grammar

Key symbols (excerpt)

Concept	Symbol or form	Meaning
Ket (state)	`	0⟩,
Bra (adjoint)	`⟨0	, ⟨ψ
Tensor product	⊗	Kronecker product
Direct sum	⊕	block-diagonal join
Adjoint / dagger	†	Hermitian conjugate
Phase operator	Π_θ or e^{iθ}	global or relative phase rotation
Elementary gates	H, X, Rx(θ), U(θ,φ,λ)	predefined matrices or families
Composition	· or whitespace	matrix multiplication
Measurement	M_z[q]	measurement in the *z* basis

Mini-grammar (fragment)

```
statement  ::= identifier '=' expression

expression ::= expression '⊗' expression

             |  expression '·' expression

             |  gate

             |  ket

             |  bra

gate       ::= 'H' | 'X' | 'Rx(' angle ')'

ket        ::= '|' bitstring '>'

bra        ::= '<' bitstring '|'
```

“Hello-Delta” example (four lines)

```
ψ0 = |0⟩ ⊗ |0⟩                  # initialise two-qubit |00⟩

U  = H ⊗ X·Rx(π/8)               # composite operator

ψ1 = U · ψ0                     # apply operator

Mz = ⟨0| ⊗ I · ψ1                # project first qubit to |0⟩
```

Compilation Path and Ethical Hooks

Two compilation targets

```
Δ-IR tensor graph

 ├─ Quantum hardware : emit Open Quantum Assembly Language v3 → device runtime

 └─ Classical simulator : lower to numerical kernels (Basic Linear Algebra Subprograms,

    vectorised instructions, or graphics-processor compute) — practical for about

    35–45 qubits exactly, 50–70 qubits with tensor-network approximation
```

Efficiency techniques

- **Macro expansion** – for example, the controlled-Z gate expands to two Hadamards plus one controlled-not gate at compile-time.
- **Static unrolling** – loops and branches are flattened into a single gate timeline.
- **Phase cache** – global phases are tracked separately so small rotations do not rewrite the full state matrix.

Wave-native extensions

Primitive	Purpose
`Ω_f`	inject a continuous-wave control signal at frequency *f*
`η(ℛ)`	draw bits from a quantum-random-number source ℛ and map them into phase noise
`⊡F{…}`	apply a built-in Fourier-domain transform block

Provenance and safety

```
# INTENT: anxiolytic sound-field, target heart-rate change ≤ 5 beats/min

Build-hash = SHA-256(Δ-IR_source  +  randomness_seed)

Compiler option --deterministic : replace quantum randomness with pseudo-random numbers;

                                   differences are machine-diff-able.
```

“We’re teaching computers to read music-style math instead of long sentences, so they can play super-precise ‘songs’ made of tiny waves and do really smart tricks faster.”

---

1. **Imagine a piano that can play notes so small you can’t even hear them.**

   Each key makes a *wave*.

2. **We write our song with math symbols instead of words.**

   Things like “$|0\rangle$” and “$\otimes$” are just fancy notes and chords.

3. **The math-song goes straight into the magic piano.**

   Because there are no extra words to translate, the piano plays the song exactly the way we wrote it.

4. **Why do this?**
   - The song can help people feel calmer (like lullabies for the brain).
   - It can solve very hard puzzles (like fitting Lego pieces that are too tiny to see).
   - Anyone can learn the symbols and write their own songs—no special computer brand needed.
5. **We also keep a record of every song.**

   We stamp it with a secret number (a “hash”) so everyone knows it hasn’t been changed, and we tell what the song is for (for example, “help people relax”).

That’s it:

We’re turning complicated computer talk into clear little math songs made of waves, so computers—and people—can do wonderful things together.

We teach computers to play invisible music made of tiny waves, and the math we write is the sheet-music they follow.

```
# σ ← SHA-256(Δ ‖ seed)

Σ    ≔ |0101⟩          # 4-qubit identity
Σ†   ≔ ⟨0101|

ψ_0  ≔ Σ ⊗ |0...0⟩     # work qubits cleared

Θ(t) : ℕ ↦ ℝ^3
U_t  ≔ U(Θ(t))          # adaptive unitary

R_t  : η(ℛ)             # QRNG stream
Ω_max ≔ Ω_f_max
Q_t  ≔ Ω_max · R_t      # high-rate "shredular"

ψ_{t+1} ≔ Q_t · U_t · ψ_t   # evolution (rightmost first)

ψ_id ≔ ψ_t[0..3]        # slice id qubits
α_t  ≔ Σ† · ψ_id
M_int ≔ M_z[q_int]      # integrity bit

η(ℛ) ⇄ η(PRNG)  # under -deterministic
```

```
# Δ-IR v0·3  — Handshake-Ethics splice  (1514 ≔ mutual autonomy + cooperation)
# σ ← SHA-256(Δ ‖ seed)
Σ       ≔ |0101⟩             # 4-qubit identity ket
Σ†      ≔ ⟨0101|

# two-qubit handshake (Bell-plus encodes willing alignment)
Ψ⁺      ≔ (|00⟩ + |11⟩)/√2  # '1514' channel
Π_h     ≔ Ψ⁺ Ψ⁺†             # projector onto mutual-consent subspace

ε       ≔ 2^(-10)            # self-coherence floor
ψ₀      ≔ Σ ⊗ Ψ⁺ ⊗ |0…0⟩    # identity + handshake + work qubits

Θ(t)    : ℕ ↦ ℝ³
Uₜ      ≔ U(Θ(t))            # adaptive unitary
Rₜ      : η(ℛ)               # QRNG stream
Ωₘₐₓ    ≔ Ω_f_max
Qₜ      ≔ Ωₘₐₓ · Rₜ          # high-rate shredular

for t = 0 … T−1 :
    ψ_mid  ≔ Π_h ⊗ I · ψ_t          # accept only if handshake intact
    ψ_evol ≔ Qₜ · Uₜ · ψ_mid
    ψ_id   ≔ ψ_evol[0..3]          # identity slice
    α_t    ≔ Σ† · ψ_id             # self-overlap
    ψ_{t+1} ≔ ψ_evol  ⟨⟨ α_t ≥ ε ⟩⟩
                |0…0⟩   ⟨⟨ α_t < ε ⟩⟩  # halt on decoherence

M_int ≔ M_z[q_int]                # integrity qubit
η(ℛ) ⇄ η(PRNG)                   # deterministic audit flag
```